ISO TC 184/SC4 Nxxx
1997-12-18

Proposed ISO TC 184/SC4 STANDING DOCUMENT

Technical Committee 184 for Industrial Automation Systems and Integration
Subcommittee 4 for Industrial Data

Guidelines for application module development
Revision 0.4

PDES, Inc. STEP Architecture and Specifications Team
Contact : David Price
dmprice@us.ibm.com
+1 301 803 2762
IBM Corporation
6700 Rockledge Drive
Mail Drop 42A1
Bethesda, Maryland 20817
USA

Contents

Page

Annexes

Figures

Tables

Foreword

The International Organization for Standardization (ISO) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

This standing document was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC4, *Industrial data*.

ISO/TC 184/SC4 standards are prepared according to guidelines put forth in the following standing documents:

— Guidelines for application module development;

— Guidelines for application interpreted construct development;

NOTE 1 - The development of new AICs is deprecated due to the advent of application modules which are intended to be the next generation AICs.

— Guidelines for application interpreted model development;

— Guidelines for the development of application protocols using application modules;

— Guidelines for the development and approval of STEP application protocols;

NOTE 2 - The development of new APs using this process is deprecated due to the advent of application modules.

— Guidelines for the development of abstract test suites;

— Guidelines for the development of mapping tables;

NOTE 3 - Clause 5.6 now contains mapping table guidelines for application modules extracted from the separate mapping table guidelines document.

— ISO/TC 184/SC4 organization handbook;

— Supplementary directives for the drafting and presentation of ISO 10303.

Introduction

ISO 10303 is an International Standard for the computer-interpretable representation and exchange of product data. The objective is to provide a neutral mechanism capable of describing product data throughout the life cycle of a product independent from any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases and archiving.

This International Standard is organized as a series of parts, each published separately. The parts of ISO 10303 fall into one of the following series: description methods, integrated resources, application modules, application interpreted constructs, application protocols, abstract test suites, implementation methods, and conformance testing. The series are described in ISO 10303-1.

The purpose of this document is to provide guidelines for the development of application modules (AMs) for use in STEP application protocols (APs). Application modules are the key component of the modularization of the initial STEP architecture. The modularization approach extends the application interpreted construct (AIC) concept of the initial STEP architecture through inclusion of the relevant portions of the AP's application reference model. The basis of the approach is understanding and harmonizing the requirements, both new and those documented in existing APs, grouping the requirements into reuseable modules, documenting the modules, and using the modules in the development of an application protocol.

The development of an application protocol modularization strategy was driven by several requirements from different sources:

— to reduce the high cost of developing an application protocol;

— to ensure the ability to implement a combination of subsets of multiple APs or to extend existing APs to meet a business need;

— to ensure the ability to reuse application software developed to support one AP in the development of an implementation of another AP with the same, or similar, requirements;

— to avoid the duplication and repeated documentation of the same requirements in different application protocols leading to potentially different solutions for the same requirements; and

— to ensure the ability to reuse data generated by an implementation of one or more APs by an implementation of one or more different APs.

This document describes the proposed content of an application module and guidelines for development of the content of the application module. The expected audience for this document includes developers of STEP application modules and application protocols as well as users of STEP application protocols who are interested in a more in-depth understanding of the origins of the structure of application protocols. This document is an adaptation of the guidance found in *Guidelines for the development and approval of STEP application protocols, Guidelines for the development of mapping tables, Guidelines for application interpreted model development,* and *Guidelines for application interpreted construct development*.

Guidelines for application module development

1 Scope

This SC4 standing document specifies guidelines for the development and documentation of application modules for use within ISO 10303 application protocols.

The following are within scope of this standing document:

— description of the content of an application module;

— guidelines for developing the content of an application module including detailed guidance on the development of scope, information requirements, mapping tables and module interpreted models.

The following are outside the scope of this standing document:

— specification of presentation information for the documentation of any portion of an ISO 10303 application module;

— detailed guidance on how to select constructs of the ISO 10303 integrated resources that map to the information requirements of an ISO 10303 application module;

NOTE 1 - This information is planned to be covered in greater detail in a forthcoming document entitled *Procedures for application interpretation*.

— guidelines for development of mapping tables for documents other than ISO 10303 application modules;

— guidelines for the use of EXPRESS in information models other than ISO 10303 module interpreted models;

NOTE 2 - The EXPRESS language is described in ISO 10303-11. This document provides EXPRESS usage guidance only in the context of module interpreted model development.

— description of how the application modules are to be used in the documentation of application protocols.

2 Normative references

The following standing documents and standards contain provisions which, through reference in this text, constitute provisions of this standing document. At the time of publication, the editions indicated were valid. All standing documents and standards are subject to revision, and parties to agreements based on this standing document are encouraged to investigate the possibility of applying the most recent editions of the standing documents and standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/TC 184/SC4 N534:1997, *Guidelines for application interpreted construct development.*

ISO/TC 184/SC4 N535:1997, *Guidelines for the development and approval of STEP application protocols.*

ISO/TC 184/SC4 Nxxx:1997, *Guidelines for the development of application protocols using application modules.*

ISO/TC 184/SC4 N533:1997, *Guidelines for the development of mapping tables.*

ISO/TC 184/SC4 N537:1997, *Supplementary directives for the drafting and presentation of ISO 10303.*

ISO 10303-11:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: Language reference manual.*

3 Definitions and abbreviations

3.1 Terms defined in ISO 10303-1

This SC4 standing document makes use of the following terms defined in ISO 10303-1.

3.1.1 application: a group of one or more processes creating or using product data.

3.1.2 application activity model (AAM): a model that describes an application in terms of its processes and information flows.

3.1.3 application context: the environment in which the integrated resources are interpreted to support the use of product data in a specific application.

3.1.4 application interpreted model (AIM): an information model that uses the integrated resources necessary to satisfy the information requirements and constraints of an application reference model, within an application protocol.

3.1.5 application object: an atomic element of an application reference model that defines a unique concept of the application and contains attributes specifying the data elements of the object.

3.1.6 application protocol (AP): a part of this International Standard that specifies an application interpreted model satisfying the scope and information requirements for a specific application.

NOTE - This definition differs from the definition used in open system interconnection (OSI) standards. However, since this International Standard is not intended to be used directly with OSI communications, no confusion should arise.

3.1.7 application reference model (ARM): an information model that describes the information requirements and constraints of a specific application context.

3.1.8 conformance class: a subset of an application protocol for which conformance may be claimed.

3.1.9 data: a representation of information in a formal manner suitable for communication, interpretation, or processing by human beings or computers.

3.1.10 implementation method: a part of this International Standard that specifies a technique used by computer systems to exchange product data that is described using the EXPRESS data specification language [ISO 10303-11].

3.1.11 interpretation: the process of adapting a resource construct from the integrated resources to satisfy a requirement of an application protocol. This may involve the addition of restrictions on attributes, the addition of constraints, the addition of relationships among resource constructs**.**

3.1.12 product data: a representation of information about a product in a formal manner suitable for communication, interpretation, or processing by human beings or by computers.

3.1.13 resource construct: a collection of EXPRESS entities, types, functions, rules and references that together define a valid description of an aspect of product data.

3.1.14 unit of functionality (UoF): a collection of application objects and their relationships that defines one or more concepts within the application context such that removal of any component would render the concepts incomplete or ambiguous.

3.2 Terms defined in ISO 10303-202

This SC4 standing document makes use of the following terms defined in ISO 10303-202.

3.2.1 application interpreted construct (AIC): a logical grouping of interpreted constructs that supports a specific function for the usage of product data across multiple application contexts.

3.3 Other definitions

For the purposes of this SC4 standing document the following definitions apply.

3.3.1 application module (AM): a reuseable collection of scope statement, information requirements, mappings and module interpreted model that supports a specific usage of product data across multiple application contexts.

3.3.2 module interpreted model (MIM): an information model that uses the common resources necessary to satisfy the information requirements and constraints of an application reference model, within an application module.

3.4 Abbreviations

For the purposes of this SC4 standing document, the following abbreviations apply.

AAM application activity model

AIC application interpreted construct

AIM application interpreted model

AM application module

AP application protocol

ARM application reference model

CC conformance class

IR integrated resource

MIM module interpreted model

UoF unit of functionality

4 Application module content

This clause provides an overview of the contents of an application module. The contents for an application module are given in figure 1 and are explained in the subsequent subclauses. The three major components of an AM are: 1) the scope and functional requirements; 2) the application reference model as a representation of the application domain information requirements; and 3) the module interpreted model that specifies the required use of the common resource.

Each application module shall have an associated abstract test suite (ATS).

Foreword
Introduction
1 Scope
2 Normative references
3 Definitions and abbreviations
4 Information requirements
- **4.1** Units of functionality
- **4.2** Referenced AM ARMs
- **4.3** ARM type definitions
- **4.4** ARM entity definitions
- **4.5** ARM rule definitions
- **4.6** ARM function definitions

5 Module interpreted model
- **5.1** Mapping table
- **5.2** MIM EXPRESS short listing

Annexes
A AM MIM short names
B Information object registration
C ARM EXPRESS-G
D MIM EXPRESS-G
E AM ARM and MIM EXPRESS listings
F Application module implementation and usage guide
G Technical discussions
H Bibliography
Index

Figure 1 - Contents of an application module

An AM starts with an Introduction which provides an overview of the technical content. For AMs which use other AMs, the Introduction shall explain the relationships between the AMs.

Clause 1 of an AM is the definition of the scope of the AM. Based on the scope, clause 4 of an AM identifies the information requirements of the AM and documents them in an application reference model. The semantics of the application reference model are specified using a documented EXPRESS schema. Based on the ARM, clause 5 specifies the selection of constructs from the common resources and identifies constraints or specializations of entities for describing the application information in an module interpreted model.

Detailed requirements for documenting an AM, including any required text, are provided in the *Supplementary directives for the drafting and presentation of ISO 10303*.

4.1 Scope

Clause 1 of an AM shall define the domain of the AM and summarize the fundamental concepts and assumptions of the scope, the functionality of the AM, and the types of information that are accommodated by the AM. A description of the functionality and information that are specifically outside the scope of the application shall be defined to clarify the domain of the AM.

This clause shall define the following characteristics of the scope of the application module:

— type of information;

— life cycle stages supported;

— uses of the information, e.g., functional processes, supported;

— discipline views supported;

— exclusions from scope for the purpose of clarification.

4.2 Normative references

All normative references shall be listed in clause 2 of an AM. The minimal required set of normative references are:

ISO 10303-1 *Industrial automation systems and integration - Product data representation and exchange - Part 1: Overview and fundamental principles.*

ISO 10303-11 *Industrial automation systems and integration - Product data representation and exchange - Part 11: Description methods: The EXPRESS language reference manual.*

ISO 10303-31 *Industrial automation systems and integration - Product data representation and exchange - Part 31: Conformance testing methodology and framework: General concepts.*

4.3 Definitions and abbreviations

Clause 3 of an AM shall include definitions of all concepts necessary to understand the Introduction, Scope, and Information requirements clauses. This clause may include concepts that are defined further in the Information requirements clause. The concept definitions provided in this clause shall be consistent with the complete definitions provided in the Information requirements clause.

This clause shall contain at least three subclauses: list of terms defined in ISO 10303-1 and used in the AM, terms defined in the AM, and abbreviations and symbols used in the AM. This clause shall not include the definitions of objects defined in the module interpreted model. This clause shall list the terms defined in other ISO standards that are necessary for understanding the AM.

4.4 Information requirements

Clause 4 of an AM shall describe the functionality and information requirements of the AM. The first paragraphs of this clause provide a high level description of the information requirements that are supported by the AM and a summary of the structure used to partition the information requirements. This clause may include a description of the types of information supported by the AM, any restrictions on the information supported, and the supported uses of the defined information. This clause shall provide all additional

information on the fundamental concepts and assumptions (initially introduced in clauses 1 and 3) which is necessary for complete understanding of the information requirements and the scope boundaries.

This clause shall include a description the fundamental concepts and UoFs of the AM and any prerequisite AMs used by the AM.

This clause shall include subclauses for units of functionality, referenced AM ARMs and ARM type definitions, ARM entity definitions, ARM rule definitions, ARM function definitions as required.

4.4.1 Units of functionality

Clause 4.1 of an AM shall specify a list of the UoFs defined in or used by the AM and the definition of each UoF defined in the AM. A UoF is a grouping of data constructs which is important in the application module. A UoF specifies the set of application objects that constitute one or more concepts of the application reference model. UoFs are a mechanism for modularising the information requirements into primary concepts. The UoFs are used to organize and summarize the functionality of the ARM. Each UoF definition shall include the scope of the UoF, a description of the function(s) that the grouping of data is intended to support, and a list of the application objects that are included in the UoF.

For a UoF defined in another application module the UoF description in the using AM shall list the application objects referenced by objects in the using AM.

NOTE - The intent is for application modules to define one UoF and possibly make use of UoFs defined within other application modules.

4.4.2 Referenced application module ARMs

Clause 4.2 of an AM shall specify the application reference models defined in other application modules that are required by the AM. This specification takes the form of documented EXPRESS USE FROM constructs.

4.4.3 ARM type definitions

Clause 4.3 of an AM shall specify the defined types necessary for the description of the application objects defined in 4.4. This specification takes the form of documented EXPRESS TYPE defintions.

4.4.4 Application objects as ARM entity definitions

Clause 4.4 of an AM shall include the definitions for all application objects and attributes declared in the AM. An application object is an atomic element of an application reference model that defines a unique application concept and contains attributes specifying the data elements of the object. Application objects are documented in the ARM by EXPRESS entity defintions.

Application objects documented as EXPRESS entities contain the definitions of simple attributes, relationship attributes and relationship constraints. Simple attributes are those which evaluate to a simple data type. Relationship attributes are those which establish a relationship with another application object.

Relationship constraints are EXPRESS INVERSE attributes which constrain the cardinality of a relationship between two application objects.

4.4.5 ARM rule definitions

Clause 4.5 of an AM shall specify the necessary rules constraining the application objects. These constraints are documented in the ARM as EXPRESS RULE defintions.

4.4.6 ARM function definitions

Clause 4.6 of an AM shall specify the necessary functions used by the ARM rules and ARM entity definitions. These constraints are documented as EXPRESS FUNCTION definitions.

4.5 Module interpreted model

Clause 5 of an AM shall specify the module interpreted model. The MIM shall be defined using the EXPRESS language and is constructed from the resource constructs (including other AMs) using the EXPRESS interfacing mechanism (USE FROM) defined in ISO 10303-11. Needs for refinement of the resource constructs arises out of the information requirements of a the particular application module domain.

4.5.1 Mapping table

Clause 5.1 of an AM shall specify the mapping table. The mapping table documents the correspondence between the information requirements and the constructs of the MIM. This mapping table shall specify a complete and unambiguous mapping between the application objects and their attributes defined in the information requirements clause and the constructs of the MIM. The mapping shows how the common resource constructs are used to meet the information requirements of the application module.

4.5.2 Short form

Clause 5.2 of an AM shall specify the MIM EXPRESS short listing. The MIM EXPRESS short listing shall consist of USE FROM statements that select common resource constructs and other AMs, AM specific declarations, and any appropriate modifications to textual material that applies to constructs interfaced into the MIM schema from the common resources. The declarations include TYPE declarations, ENTITY declarations that create subtypes of resource entities, and any necessary RULES, FUNCTIONS, and PROCEDURES that are required to satisfy the information requirements. Any declarations of entities, rules, functions, and procedures specific to the AM are fully documented in the MIM EXPRESS short listing. Textual modifications include:

— clarification of application specific interpretation of the meaning of a generic entity definition;

— clarification of application specific interpretation of the meaning of one or more attributes;

— specification of application specific informal propositions;

— specification of all associated global rules defined in the AM;

— addition of application specific examples and notes.

4.6 Annexes

A MIM short names (normative)

This annex shall contain a correspondence list between the entities used in the MIM and the short names. This list is derived from the short names specified in the common resources together with the short names for entities introduced in the AM.

B Information object registration (normative)

This annex shall specify the information object identifiers for the application module. This shall include identifiers for the AM document and for the MIM schema

C ARM EXPRESS-G (informative)

This annex shall contain the EXPRESS-G representation of all interface specifications and all types and entities defined in the ARM. This representation shall be documented in accordance with annex D of ISO 10303-11 and the *Supplementary directives for the drafting and presentation of ISO 10303 (STEP)* [9].

D MIM EXPRESS-G (informative)

This annex shall contain the EXPRESS-G representation of all interface specifications and all types and entities defined in the MIM. This representation shall be documented in accordance with annex D of ISO 10303-11 and the *Supplementary directives for the drafting and presentation of ISO 10303 (STEP)* [9].

E MIM and ARM EXPRESS listing (informative)

This annex shall contain a disk with the entire MIM and ARM EXPRESS short listing without comments.

F Application module implementation and usage guide (optional and informative)

This annex, if provided, contains informative guidance on implementing and using the AM. This annex provides guidance to two different audiences, i.e., implementors and end users of AM compliant implementations. Example information descriptions that are supported by the AM and the corresponding AM exchange files may be included in this annex. If exchange files are included in this annex, the annex should explain the primary data structures and the logic and meaning of the values used in the exchange file.

G Technical discussions (optional and informative)

This annex, if provided, contains a summary of relevant technical discussions and the resolution of issues raised during the development of the AM. This annex provides background information for potential

users of the AM and for developers of similar or related AMs. The material given should not cast doubt or self justify. Only material which supports the normative text shall be given.

H Bibliography (informative)

This annex lists all informative references relevant to the AM. At a minimum, it shall contain references to the *Supplementary directives for the drafting and presentation of ISO 10303*.

5 Specification of application module content

This clause specifies the requirements for developing an AM. An AM shall be developed and reviewed incrementally as components of the AM documentation are completed. The major objective of the incremental development is for AM integration planning. A review of the AM's scope and information requirements, prior to developing the MIM, allows international consensus on the detailed requirements to be developed. It also allows identification of requirements common with other applications for the purpose of AM integration planning.

5.1 Scope development

The first phase of developing an AM is the definition of its scope and information requirements. Definition of the scope and information requirements begins with the formulation of a statement of the application module functional requirements. This statement may define the type(s) of information, the life cycle stages supported, the data application(s), and the use of the data within the application(s) targeted for the AM. The detailed scoping and information requirements definition shall proceed from this statement.

The scope and requirements identify the primary concepts and relationships to be supported by the AM. The AM's scope and information requirements shall be carefully defined and documented. The AM scope statement may include a summary of the type(s) of information, the application processes, the types of data, and the discipline views of the data that are within scope. For clarification, the scope statement may also identify the type(s) of information, the type(s) of products, the application processes, the types of product data, and the discipline views of the data that are outside of the scope.

5.1.1 Scope refinement

The refinement of the scope of an AM includes identifying and separating concepts to maximise reuse of the AM and identifying mandatory relationships with other AM's.

The AM should define a unique and distinct set of Application Objects relating only to the concept being represented. It should contain no Application Objects that are defined by another AM, that is representing another concept.

Vocabulary will typically change from concept to concept. The description of the AM should not contain sentences that move from one concept to another. The use of words such as 'and', 'with' or 'also' in either the name or description of an AM is an indication that there is more than one concept being represented. Unless

it can be demonstrated that the concepts being represented are inseparable in all applications, serious consideration should be given to separating those concepts and defining an AM for each.

NOTE - The scope AM should not mix fundamental concepts such as presentation with representation.

5.2 Information requirements specification

When the detailed scope and functional requirements have been defined, the information domain of the AM shall be defined by developing the application reference model (ARM). The ARM shall be documented using EXPRESS and shall include an EXPRESS-G presentation. The ARM shall describe fully the data needs of the application module, using the potentially harmonized terminology of the application domain(s).

The ARM documents the required data and relationships of the AM. The graphical presentation of the ARM, i.e., EXPRESS-G, aids the understanding and review of the information requirements and definitions. The ARM diagrams shall be at a detail level sufficient to present the requirements in a manner that it is understandable to an application domain expert. The information requirements shall be modelled only to the level necessary to convey the information that is important from the application experts' point of view. An ARM shall be sufficiently detailed so that the selection and interpretation of the common resources can be done accurately.

A mechanism for modularizing the scope of an industry domain into manageable constructs is to define units of functionality. A UoF is a collection of application objects and assertions that conveys one or more well-defined concepts within the context of an ARM. A UoF may result in one or more AM's and an AM may include or use more than one UoF. However, to realize the most reusability of an application module the scope of application modules that define multiple UoFs should be reviewed for potential refinement (see 5.1.1). A UoF usually supports an application function or process. UoFs are used to organize and summarize the functionality of the ARM. For example, if a geometric modelling application has a requirement for wireframe geometry, then a UoF may be defined which provides a grouping of those application objects in the ARM which are intended to support geometric modelling using wireframe geometry.

When two or more AMs have equivalent UoFs or common information requirements, a new AM shall be created, the same interpretation of the integrated resources shall be used in the new AM, and the new AM shall be used by the AM's with common requirements. During the development of an AP or an AM it may be discovered that an existing AM supports a superset of the requirements for the AM in development. In this case, new AMs should be developed separating the required subsets from the existing AM. The original AM may then be updated to use the new AMs without changing the scope of the existing AM. Through this process the scopes of the catalog of AMs is incrementally refined as new or differing requirements are discovered.

5.2.1 Information requirements documentation

Clause 4 of the AM shall include a high level description of the information requirements, a summary of the structure used to define the partition of the information requirements defined by the AM, and subclauses for specifying UoFs, referenced AMs and the AM ARM components. The description of the information requirements shall be sufficient to prepare the reader for the material in the subclauses.

The information requirements shall be defined in prose. The referenced AMs and the ARM components including application objects, relationship attributes and constraints shall be defined as a single documented EXPRESS schema. The elements listed within each subclause shall be ordered alphabetically. The UoFs, ARM schema, ARM types, application objects, ARM rules and ARM rules shall have unique names, i.e., no application elements shall share the same name.

The documentation for an AM's ARM and information requirements includes the following components as required. The clause referenced listed assume at least one of each component is included. Should there be no ARM types, functions or rules these clauses may be omitted in the AM and the clause numbering shall change accordingly.

1. units of functionality

Units of functionality shall be defined in 4.1 of the AM. This subclause provides a list of the UoFs used or defined in the AM. For UoFs defined in the AM, a description of the functions that each UoF supports, and the list of application objects included in the UoF is specified. For UoFs used fromanother AM, the name of the AM in which which the UoF is defined shall be included in the list. Application objects defined in another AM specifically referenced by application modules defined in the AM shall be listed under the description of the UoF in the other AM in which they are defined.

2. referenced application module ARMs

The referenced application module ARMs shall be defined in clause 4.2 of the AM. This specification takes the form of documented EXPRESS USE FROM constructs. Each AM ARM shall be used in its entirety. Should the AM no use any other AM this fact shall be stated in this clause. The ARM uses the terminology of the application domain but should not be so specific as to prohibit reuse of the AM in multiple APs.

3. ARM type definitions

The ARM defined types shall be defined in clause 4.3 of the AM. This specification takes the form of documented EXPRESS TYPE definitions.

4. ARM entity definitions

The application objects shall be defined in 4.4 of the AM and are documented as ARM EXPRESS entities. Every EXPRESS entity in the ARM shall be considered an application object. Each application object which exists in the ARM shall appear in the mapping table.

5. ARM rule definitions

The ARM rules shall be defined in clause 4.5 of the AM. These rules specify the necessary constraints on the application objects. These constraints are documented in the ARM as EXPRESS RULE definitions.

6. ARM function definitions

The ARM functions shall be defined in clause 4.6 of the AM. These functions specify the necessary functions used by the ARM rules and ARM entity definitions. These functions are documented as EXPRESS FUNCTION definitions.

5.2.2 EXPRESS ARM documentation

The ARM EXPRESS definitions shall be specified as follows:

— The ARM type definitions shall appear in alphabetical order.

— Each application object shall be stated in the ARM entity definitions and each ARM entity definition shall represent an application object.

— Each attribute whose data type is either a base data type or a defined data type which is a SELECT data type with a select list that does not contain entity types or, recursively, other SELECT types with select lists that do not contain entity types shall be stated as an attribute of that entity in the ARM entity definition.

— Each attribute whose data type is an aggregate with a type that is either a base type or a defined type which is a SELECT data type with a select list that does not contain entity types or, recursively, other SELECT types with select lists that contain entity types shall be defined as an attribute in the ARM entity definition, with the cardinality specified in the definition.

— Each relationship attribute whose data type is an aggregate of either an entity type or a SELECT type with a select list that contains either entity types or other select types with select lists that contain entity types shall defined as an attribute in the ARM entity definition with the cardinality defined by the aggregate bounds.

— Each relationship attribute whose data type is an entity type shall be defined as an attribute in the ARM entity definition.

— Each relationship attribute whose data type is a SELECT data type with a select list that contains entity types or other select types with select lists that contain entity types shall be defined as an attribute of the ARM entity definition.

— Each constraint on the cardinality of a relationship attribute whose data type is an entity type shall be defined as an INVERSE attribute in the referenced ARM entity definition.

— The attributes of the ARM entity definitions shall be ordered such that the non-relationship attributes are preceed the relationship attributes. The non-relationship attributes shall appear in alphabetical order. The relationship attributes shall appear in alphabetical order. The INVERSE attributes shall appear in alphabetical order.

— The ARM rule definitions shall appear in alphabetical order.

— The ARM function definitions shall appear in alphabetical order.

5.3 Module interpreted model

After the information requirements have been documented, interpretation of the information requirements results in the development of the MIM. Interpretation is based on the foundation laid by the review of the scope and requirements documented in the AM. The objectives of interpretation are:

— to select the required SC4 common standardized resource constructs to satisfy ARM requirements;

— to map ARM constructs to the selected resource constructs; and

— to create the additional EXPRESS constructs and constraints needed to satisfy ARM requirements and to specify the MIM short listing.

5.3.1 Select resource constructs

During interpretation, appropriate resource constructs are selected to meet the requirements for the ARM constructs. The selection process often involves further discussions or clarifications of the ARM constructs; these may include review of the intended information requirements, the usage of the information, or description of real-world usage scenarios of the application module. The result of these discussions is the specification of one or several resource constructs which satisfy the requirements of the ARM construct, along with any needed constraints.

The integrated resource constructs are designed for generic use by all AMs. In the selection process, the best entity for an ARM requirement may have attributes that are not supported by requirements in the ARM. As long as the additional attributes have underlying types that are base types, this is not considered a problem. For example, many resource entities include required attributes such as name and description. These attributes have the base type STRING. Where there is no data to support the instantiation of these attributes, the developers of an application module may choose to document recommended values for these attributes in the "Module implementation and usage guide" annex of the application module.

5.3.2 Short form specification

An MIM is constructed from the common resources, including AMs, through the use of an inter-schema reference specification; this MIM schema is called the short listing. The short listing consists of two parts. The first part contains the interface specification which provides the relationship between the resources and the MIM. The second part defines the unique MIM constructs that refine or specialize the usage of the integrated resources.

The development of the short listing consists of two tasks:

— specify the interface to the resources; and

— specify MIM unique constructs and constraints.

This clause of this document describes the development of the short listing from a functional perspective. The remainer of clause 5 of this document provides more detail on the use of EXPRESS to document the short listing.

5.3.2.1 Establish a formal interface to the resources

The EXPRESS short listing reflects the selection of the resource constructs through a formal interface specification method. This interface is accomplished through the EXPRESS USE FROM specification described in clause 11 of ISO 10303-11. USE FROM is the interface mechanism that is employed to specify the relationship between the resources and the MIM. The information provided in 5.3.4 of this document will provide more details on how to use the EXPRESS language to completely specify the interface to the resources.

5.3.2.2 Develop unique MIM constructs

The MIM short listing also specifies all constructs that are new and unique to the MIM, which may include entities, attributes, type definitions, local and global rules, and functions. Only two classes of new constructs are allowed in an MIM; those that:

— complete and assign a concept definition; and

— constrain a generic concept.

5.3.2.2.1 Concept completion and assignment

In the ISO 10303 integrated resources, there are a number of template structures that have been used to specify product data management resources, such as the construct specified in annex E of ISO 10303-41. These template structures are semantically incomplete by themselves; they are designed to be used to specify structurally similar though semantically dissimilar concepts in a standard and consistent manner. The template entity is completed by an explicit and unique assignment of semantics in the MIM. The explicit assignments are made between the management concepts (e.g., approval or organization) and the items that require the management information. Details on how the new MIM constructs which complete the assignment template structure are documented in the MIM short listing are provided in 5.3.4.2.3 of this document.

5.3.2.2.2 Constrain generic concept

Standardized common resource constructs are generic in nature and designed to be shared by multiple application contexts. Several AMs may require the same generic concept that a resource construct represents, but each may constrain the generic concepts in order to represent a specific usage within the AM domain. New MIM constructs may be created (through subtyping) to:

— constrain a generic concept;

— constrain the relationship between generic concepts; or

— create multiple, specialized concepts of the generic concept, through specification of different constraints representing different usages of the same generic concept under specific circumstances within the domain of the AM.

5.3.3 Complete short form schema

Once all entities and types are interfaced into or defined in the MOM, global rules may also be defined in the MIM to constrain an entity, relationship or attribute.

EXAMPLE - Global rules are often written to constrain an entity that was explicitly interfaced from being independently instantiated.

The constructs from the integrated resources are pruned in the MIM (see 5.1.2 of *Guidelines for application interpreted model development*). There may be subtypes and items of select lists that appear in the integrated resources that are not imported into the AM. Constructs are eliminated from the subtype tree or select list through the use of the implicit interface rules of ISO 10303-11. References to eliminated constructs are outside the scope of the MIM.

5.3.4 EXPRESS usage guidelines for MIMs

In the previous clause, the process of interpretation was presented from a functional viewpoint. This clause will describe the use of the EXPRESS language for developing the MIM EXPRESS short listing, and provide additional details that describe the semantics of a particular usage of EXPRESS.

5.3.4.1 Schema interface

The need for an MIM to establish a formal relationship to the integrated resources through the EXPRESS USE FROM keyword is described in 5.3.2.1. USE FROM is the only interface mechanism that shall be employed in an MIM.

The USE FROM keyword is employed to interface named data types and entire schemas into another schema. In an MIM EXPRESS short form, if the interface is to a construct defined in the integrated resources, the desired construct is individually named in the interface specification for the schema in which the construct is defined. If all constructs from an integrated resource schema are desired, the entire schema may be interfaced by providing only the schema name in the interface specification. If the interface is to an AM, only the AM schema name is provided. The AM must be used in its entirety; the use of subsets of an AM is not allowed.

The objective in specifying the interface to the resources is to specify the minimum set of constructs in USE FROM statements that:

— satisfies the visibility requirements for creation of unique MIM constructs;

— results in the required content of the MIM; and

— satisfies the requirements for independent instantiability.

In order to be able to accomplish this objective, the AM development team must understand the interface mechanism as described in ISO 10303-11, know which constructs from the integrated resources are required in the MIM, and know which AMs are required in the MIM.

5.3.4.2 Entity type specialization

5.3.2.2 describes the requirements for the declaration of unique MIM constructs during the development of the MIM EXPRESS short listing. These constructs are developed in a strictly controlled manner to accomplish strictly defined objectives. Entities are created in the MIM only to achieve one of three objectives: constraint localization, attribute definition specialization, and concept completion and assignment. Concept completion and assignment also requires the specification of a SELECT type. In the development of an MIM, this is the only case where an MIM-specific defined type is specified.

5.3.4.2.1 Localization of constraints

During MIM development, there may be requirements to specify constraints on an entity that differ depending on its usage in different structures. This type of constraint is called an entity behavioral constraint. In order to specify this type of constraint, the entity is interfaced explicitly into the MIM schema from the integrated resources or an AM. There are two methods for representing the constraint. Either a new subtype of the entity is be created to represent the concept for which the constraints are defined or global rules are defined to constrain attribute values representing the required constraint from the ARM. Using global rules is the first method to consider to represent these constraints. Subtyping should be employed when the complete set of subtypes for all possible domains can be determined.

An example of the use of entity behavioral constraints consists of an application requirement defined in an ARM for two different types of the `product_definition` entity. One type of `product_definition` entity is always a component in an assembly and another type of `product_definition` is never a component in an assembly. This is a behavioral constraint on the `product_definition` that could be implemented with two subtypes or with constrained attribute values. In the case where subtypes are created, the first subtype could contain a constraint that says it must always be used in the `product_definition_relationship` entity as the `related_product_definition`[1]. The second subtype of `product_definition` could contain a constraint specifying that it shall never be used in the `product_definition_relationship` entity. In the case where attribute values are constrained, a global rule could be written specifying that when the description attribute of product_definition has a value of component, the product_definition must always be used in a product_definition relationship as the related_product_definition.

Entity behavioral constraints are specified as local rules. In the case that subtypes are created, the EXPRESS USEDIN function is employed to gain access to the other entities that reference the particular entity which needs to be constrained in the MIM. Each subtype definition allows the MIM schema to specify different uses of the generic concept (as defined in the STEP integrated resources) for different purposes (as defined in an ARM).

---

[1] `Product_definition_relationship` and `product_definition` are defined in ISO 10303-41.

In addition to constraints on the usage of a particular entity, there may be a need to specify differing, and often conflicting, constraints on an entity or an entity's attributes depending on its usage in the reference path of a generic entity interfaced to the MIM. In this case, subtypes of the referencing entity shall be created in order to establish a context for the constraint and to specify the constraint on the referenced entity. This type of constraint is called a referential integrity constraint.

For example, referential integrity constraints would be used to support an application requirement for two separate uses of the `polyline` entity defined in ISO 10303-42. Let us assume for the case of this example, that the requirements for the usage of the `polyline` are differentiated by the fact that there are two different mathematical methods for describing the shape of something. One representation requires polylines to contain exactly two points for the representation of line segments. The other representation has a requirement for polylines to contain more than two points and line segments defined by trimmed curves with underlying lines as the basis curves. This example will use the entities defined in ISO 10303-41, ISO 10303-42 and ISO 10303-43.

Since the `polyline` is constrained differently based on its usage in the particular method, if both methods are required in a single AP, the constraints on use of the `polyline` entity must be localized. Constraint localization is accomplished by defining an entity in the MIM schema that is a subtype of a resource entity to define a scope for the constraints. To localize the constraints in this example, two subtypes of the `shape_representation` entity from ISO 10303-41 need to be created where the applicable constraints for the `polyline` are specified in the two different representation subtypes. One of them is to create a scope for the mathematical method in which polylines are defined only by two points; and the other is to create a scope for the mathematical method in which polylines are defined by more than two points. The entity definition for the subtypes shall contain an explanation of the purpose of the constraints. The `polyline` entity is referenced by an attribute of the geometric_set entity, which is, in turn, referenced by the representation structure from ISO 10303-43. The constraints on `polyline` will be specified as referential integrity constraints on the `representation_items` (inherited by `shape_representation` entity from the `representation` entity in ISO 10303-43) that are of type "polyline" within the contents set of the `geometric_set` that is in the set of items in the `representation` entity. In one `shape_representation` subtype, the size of the set of points that define the polyline is constrained to two elements, and in the other `shape_representation` subtype the size of the set of points that define the polyline is constrained to be more than two. Each MIM entity (the created subtypes of the `shape_representation` entity), therefore, defines a context within which conflicting constraints on the polyline may exist within the MIM schema.

Referential integrity constraints are written as local rules that use the EXPRESS TYPEOF function to identify the appropriate traversal through the reference path in specifying the constraint on the subtype and gain access to the attributes that ultimately need to be constrained. In the example, each subtype allows different constraints to be placed on a single attribute of a single entity depending on the reference path by which those attributes are reached.

5.3.4.2.2 Specialization of attribute definitions

The creation of subtypes of the interfaced entities from the integrated resources or AMs enables more specific attribute definitions to be given when the generic definition is not sufficient to satisfy the ARM requirements. This situation most often occurs when there is an ARM requirement for relationships

defined between two entities that play specific roles within those relationships. There are two practices which may be used in this case, depending on the application requirements.

The first practice concerns attribute naming. If there is a requirement that an attribute have a specific name based on the ARM, then a subtype entity is created in the MIM and a derived attribute is specified which names the attribute inherited from the supertype entity. Derived attributes may be used, for example, to specialize the generic `product_definition_relationship` entity when an application requirement states that the relationship is prioritized: one `product_definition` is first priority and the other is second priority. The attributes names `first_priority` and `second_priority` have a very specific application meaning and the development team has defined those roles specifically within the ARM requirements. The MIM would then contain an entity such as:

```
*)
ENTITY priority_product_definition_relationship;
SUBTYPE OF (product_definition_relationship);
DERIVE
  first_priority : product_definition :=
   SELF\product_definition_relationship.relating_product_definition;
  second_priority : product_definition :=
   SELF\product_definition_relationship.related_product_definition;
END_ENTITY;
(*
```

The second practice is used if there is no application requirement for a specific attribute name. If the definition of an inherited attribute must be specialized in the MIM and there is no application requirement for an attribute name to be given, a subtype may be created and additional textual definitions are created in the MIM for inherited attributes. This entity would then be used to represent the semantics given in the textual definitions of the inherited attributes.

An example of this second practice may be found in ISO 10303-203. The entity `supplied_part_-relationship` is declared as a subtype of the generic `product_definition_relationship` entity. Since there are no application requirements for specific attribute names, the definitions are refined to say that the `related_product_definition` is to be interpreted as that `product_definition` that is supplied by an outside organization and the `relating_product_definition` is the `product_-definition` that is internal to the owning organization.

5.3.4.2.3 Concept completion and assignment

Concept completion and assignment is a process allowed in both application modules and in application protocols.

> NOTE - This is process is expected to be allowed only in application modules in the future when the partial select type capability being defined in the next edition of EXPRESS can be used. Alternatively, a non-standard process might be utilized in the AM or AP EXPRESS processing to address the issue of completion and extension of the SELECT types necessary for concept completion and assignment.

Subtypes of interfaced entities may be created for the completion and assignment of generic concepts. The function of concept completion and assignment is described in 5.3.2.2. The assignment of a generic concept is the only time that an explicit attribute may be added to a subtype declared in an MIM schema. In order to assign the generic constructs to the appropriate entities as defined in the ARM requirements, two constructs are created. The first construct is a SELECT type that contains all of those entities or other SELECT types that may have the concept assigned to it. The second construct is an entity which is a subtype of the interfaced entity representing the generic concept. This entity shall contain a single attribute which references a SET of the SELECT type that was defined previously. A SET is used here so that all concepts that are the same are able to be assigned to different instances of entities that share the information. The following example illustrates the use of the concept completion and assignment technique in an MIM. The `items` attribute of the new SUBTYPE `applied_date_assignment` that references the new SELECT type `date_assigned_items` illustrates the only case where it is allowable to add an attribute in a subtype.

```
*)
SCHEMA resource_example_schema;

  REFERENCE FROM date_schema (date, date_role);

  ENTITY date_assignment;
  ABSTRACT SUPERTYPE;
    role : date_role;
    assigned_date : date;
  END_ENTITY

END_SCHEMA; --resource_example_schema

SCHEMA concept_completion_example_schema;

  USE FROM partial_product_definition_schema
    (product,
     product_definition_formation);

  USE FROM resource_example_schema (date_assignment);

  TYPE date_assigned_items = SELECT -- SELECT type definition
    (product,
     product_definition_formation);
  END_TYPE;

  ENTITY applied_date_assignment; -- ENTITY subtype definition
  SUBTYPE OF (date_assignment);
    items : SET [1:?] OF date_assigned_items;
  END_ENTITY;

END_SCHEMA; --concept_completion_example_schema
(*
```

The new subtype entity declaration may also be combined with the localization of constraints practice which would enable the specification of either behavioral or referential integrity constraints in the WHERE clause of the entity. An example of this is provided in the following example schema; date assignments are coordinated with the value of the role attribute. The function `date_time_-correlation` says, for example, that a date with the role of "creation date" must be assigned to a product_definition entity. Again, the addition of attributes in an MIM-defined subtype is allowed only for the assignment template structures in the STEP integrated resources or AMs; these ABSTRACT SUPERTYPE entities are incomplete by definition and may be completed in the MIM.

```
*)
SCHEMA role_correlation_example;

TYPE date_time_item = SELECT  -- these entities not provided in this schema
  (product_definition,
   change_request,
   start_request,
   change,
   start_work,
   approval_person_organization,
   contract,
   security_classification,
   certification);
END_TYPE; -- date_time_item

ENTITY role_correlation_date_and_time_assignment
  SUBTYPE OF (date_and_time_assignment);
  items : SET [1:?] OF date_time_item;
WHERE
  WR1: date_time_correlation(SELF);
END_ENTITY; -- role_correlation_date_and_time_assignment

ENTITY date_and_time;
  date_component : date;
  time_component : local_time;
END_ENTITY; -- date_and_time

ENTITY date_and_time_assignment
  ABSTRACT SUPERTYPE;
  assigned_date_and_time : date_and_time;
  role                   : date_time_role;
END_ENTITY; -- date_and_time_assignment

ENTITY date_time_role;
  name : label;
END_ENTITY; -- date_time_role
```

```
FUNCTION date_time_correlation
  (e : role_correlation_date_and_time_assignment ) : BOOLEAN;
  LOCAL
    dt_role : STRING;
  END_LOCAL;
    dt_role := e\date_and_time_assignment.role.name;
  CASE dt_role OF
    'creation_date'         : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                'ROLE_CORRELATION_EXAMPLE.' +
                                'PRODUCT_DEFINITION'
                                IN TYPEOF (x)))
                                THEN RETURN(FALSE);
                              END_IF;
    'request_date'          : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                SIZEOF (
                                ['ROLE_CORRELATION_EXAMPLE.CHANGE_REQUEST' +
                                'ROLE_CORRELATION_EXAMPLE.START_REQUEST'] *
                                TYPEOF (x)) = 1))
                                THEN RETURN(FALSE);
                              END_IF;
    'release_date'          : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                SIZEOF (
                                ['ROLE_CORRELATION_EXAMPLE.CHANGE' +
                                'ROLE_CORRELATION_EXAMPLE.START_WORK'] *
                                TYPEOF (x)) = 1))
                                THEN RETURN(FALSE);
                              END_IF;
    'start_date'           : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                SIZEOF (
                                ['CONFIG_CONTROL_DESIGN.CHANGE' +
                                'ROLE_CORRELATION_EXAMPLE.START_WORK'] *
                                TYPEOF (x)) = 1))
                                THEN RETURN(FALSE);
                              END_IF;
    'sign_off_date'         : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                'ROLE_CORRELATION_EXAMPLE.' +
                                'APPROVAL_PERSON_ORGANIZATION'
                                IN TYPEOF (x)))
                                THEN RETURN(FALSE);
                              END_IF;
    'contract_date'         : IF SIZEOF (e.items) <>
```

```
                                SIZEOF (QUERY (x <* e.items |
                                'ROLE_CORRELATION_EXAMPLE.CONTRACT'
                                IN TYPEOF (x)))
                                THEN RETURN(FALSE);
                              END_IF;
    'certification_date'    : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                'ROLE_CORRELATION_EXAMPLE.CERTIFICATION'
                                IN TYPEOF (x)))
                                THEN RETURN(FALSE);
                              END_IF;
    'classification_date'   : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                'ROLE_CORRELATION_EXAMPLE.' +
                                'SECURITY_CLASSIFICATION'
                                IN TYPEOF (x)))
                                THEN RETURN(FALSE);
                              END_IF;
    'declassification_date' : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                'ROLE_CORRELATION_EXAMPLE.' +
                                'SECURITY_CLASSIFICATION'
                                IN TYPEOF (x)))
                                THEN RETURN(FALSE);
                              END_IF;
    OTHERWISE : RETURN(TRUE);
  END_CASE;
  RETURN (TRUE);
END_FUNCTION; -- date_time_correlation

END_SCHEMA; -- role_correlation_example
(*
```

5.3.4.3 Use of global rules

When an application requirement results in a constraint on every use of an interfaced entity or attribute of an interfaced entity, a global rule is specified in the MIM. Global rules are also used to specify constraints on the relationships among two or more entities, on attribute values of entities interfaced into the MIM and on the use of an interface entity for a specific purpose when one or more roles for the use of the entity can be identified.. Constraints may be specified as global rules in an MIM to serve several specific functions. These functions are described in the following subclauses. It is recommended that AM development teams maintain a matrix that maps requirements from the AM to the global or local rules that implement the requirements in the MIM.

5.3.4.3.1 Supertype constraint

A supertype constraint is one that constrains the relationships among interfaced entities in the same subtype/supertype tree. These constraints are in addition to any constraints applied through the specification of the interface statements (i.e., dependence). There are many uses for this type of constraint; only two uses are described here.

A supertype constraint, for example, may make the instantiation of a subtype mandatory for a particular supertype entity. The requirement for a rule of this type will arise when an entity is 1) explicitly interfaced to be subtyped, or 2) is implicitly interfaced via attribute reference as well as subtype reference. If the ARM requirement is only for the new subtype entity, then a rule is required to prohibit the instantiation of the interfaced supertype entity. See annex C for a template for a mandatory subtype global rule.

Rules for constraining supertype relationships may also be used to limit the combinations of subtypes of a single supertype entity in order to define the appropriate set of complex entity data types allowed within the MIM that satisfy the requirements specified in the ARM.

5.3.4.3.2 Cardinality constraint

A cardinality constraint is one that constrains the relationship between two interfaced entities by limiting the number of instances of one entity type that may be related to instances of the other. Explicit cardinality is specified in the referential direction of the relationship (i.e., from the entity with the attribute to the entity that is the data type of that attribute). When an INVERSE attribute does not exist to constrain the cardinality of the referenced entity, the cardinality defined is, by default, zero, one or many. These cardinalities may be constrained using a cardinality constraint in the MIM. See annex C for a template for a global rule which constrains cardinalities of referenced entities.

The integrated resources, for example, always model the existence dependency of one entity with respect to another by the referential direction of the relationship. That is, the dependent entity always references the independent entity so that an instance of the dependent entity requires the existence of an instance of the independent entity. In the ARM, two entities might be interdependent. That is, each entity requires a single instance of the other in order to model a complete concept. In this case, a cardinality constraint would be necessary to constrain the inverse cardinality to be exactly one rather than the default zero, one or many.

5.3.4.3.3 Referential integrity constraint

Global rules can be used to specify referential integrity constraints as defined in 5.3.4.2.1. This type of constraint is one that constrains valid reference paths for all instances of an interfaced entity. The paths that are constrained may be combinatorial in nature. These combinatorial constraints are ones where a single entity may be required to be instantiated through two or more distinct paths in order to completely satisfy an ARM requirement. This type of constraint may also constrain an attribute value that is reached via a single reference path for all instances of a referencing entity.

5.3.4.3.4 Attribute domain constraint

An attribute domain constraint is one which constrains the value of an attribute in instances of a particular entity. This constraint is used to constrain the values, for example, of an attribute of type STRING in an interfaced entity to correspond to values of an enumeration as defined in ARM requirements. In the design of

the integrated resources, the use of enumerated types was limited to preserve the generic nature of the integrated resources. Instead of referencing enumerated types, integrated resource entity attributes reference base types or defined types where the underlying type is a base type (i.e., INTEGER, STRING etc.). When an application context as defined in an ARM is used to determine requirements for specific structures, enumerations are considered specific requirements. A domain constraint specifies the legal values of the simple data types that correspond to the enumeration values and the standard interpretation of those values for that application context. See annex C for a template for a global rule which restricts attribute values.

An attribute domain constraint may be used to place limits on the values of attributes of an entity. For example, if an entity has an attribute that is an INTEGER, and an ARM specifies a requirement that the INTEGER value must be greater than 1, the domain constraint is specified on the entity that contains that attribute to declare that constraint.

An attribute domain constraint may also be used to constrain the use of instances of an entity to be those plying some role in a particular usage of that entity type. This type of constraint replaces subtyping and adding local constraints to limit the valid relationships in which the entity may paricipate.

5.3.4.4 Use of functions

In the specification of rules, EXPRESS functions may be used in order to make the specification of the rule simpler. Due to the recursive nature of many of the references in the common resources, a constraint may need to be defined recursively. Additionally, many rules may need to use the iterative and logical capabilities of the executable statements defined in EXPRESS in order to specify complex constraints on the interactions among different entities. This type of complex interaction will usually require that a function be defined in the MIM in order to support the specification of a constraint. In these cases, new functions may be defined in an MIM short listing to be used by a rule.

5.4 Define new MIM constructs

The AM document must include complete definitions of all new MIM constructs found in the MIM EXPRESS short listing. It may also include textual definitions and descriptions from the interfaced resource constructs that have been further specialized to suit the established application context of the AM. Guidance for developing and formatting these definitions is found in *Supplementary directives for the drafting and presentation of ISO 10303*.

5.5 Produce short names table

Short names are required for each entity in the MIM EXPRESS short form schema. An explanation of short names is found in *Guidelines for the development and approval of STEP application protocols*. The short names are generated by a software tool used by the SC4 Secretariat at NIST. Prior to publication of any part containing an EXPRESS schema, the schema shall be sent to NIST for short name generation. Guidance on documenting short names in the AM is given in the *Supplementary directives for the drafting and presentation of ISO 10303*.

5.6 Mapping table specification

The mapping table is a pivotal component of an AM. The mapping table specifies the relationship between the information requirements as specified in the ARM and the resource constructs that satisfy those requirements in the module interpreted model. The remainder of this subclause discusses how interpretation results are captured within the mapping table.

Initially, the left-hand column of the mapping table is populated with the application objects, non-relationship attributes and relationship attributes. During interpretation, resource constructs that satisfy the information requirements are selected and documented in the table.

There is rarely a one-to-one mapping between constructs in the ARM and MIM because the ARM and the SC4 common resources were defined under different contexts. During interpretation, constructs from the resources are selected for mapping to the ARM constructs. A particular ARM object may map to one resource construct or more than one resource construct in an AND situation, an OR situation or a combination of the two. The details of the mapping, including the circumstances under which each mapping applies shall also be identified and documented.

The mapping table specifies the data reference path for certain mappings. A data reference path is a directed sequence of (MIM) entities that are linked through attribute relationships or subtype/supertype relationships. Reference paths are specified in the mapping table only when the mapping of the application object from the ARM to the resource constructs creates one of the following circumstances:

— A new entity is defined in the MIM as a subtype of a resource entity. Constructs that represent completely new concepts cannot be independently specified in the MIM, but must be derived from existing resource constructs. The MIM extends the definitions from existing resources through specialization. All new entities introduced in the MIM are subtypes of resource entities with refined definitions or added constraints. New entities are created in order to add constraints to an existing concept (represented by a construct from the resources). These extensions to the generic definitions apply only within the particular application context defined in that AM. The reference path for new entities introduced in an MIM must include a resource supertype entity.

— An ARM attribute and the ARM entity do not map to the same entity. An ARM attribute may be mapped to either a resource entity or an attribute of a resource entity. When the ARM entity to which this attribute belongs is mapped to a different resource entity, a reference path specification is required. The reference path specification provides the entire access path which begins at the MIM element corresponding with the ARM entity to which the ARM attribute belongs, and ends at the MIM element corresponding with the ARM attribute. When the ARM attribute maps to an attribute of the MIM entity to which the ARM entity maps, no reference path is required.

— An attribute of an MIM entity must be assigned a specific value. The mapping of an ARM entity to a resource entity may require that an attribute (or attributes) of the resource entity have a specific value. In the mapping, the value of the attribute(s) must be restricted or assigned in the specification of the reference path. A reference path will show the mapped MIM entities and attributes and the attribute values.

— A relationship exists between two ARM entities that map to different resource entities. The relationship is mapped as a reference path between MIM entities. The reference path shows the entire data access path between the mapped resource constructs.

— A constraint on the mapping must be documented. Such mapping rules may be included in reference paths that are required for one of the above reasons, or a path may be specified for the sole purpose of including a mapping rule.

Each information requirement of an AM is mapped to one or more constructs in the MIM. Each of these mappings is documented on a single row of the mapping table. The mapping table is organized into sub-tables by unit of functionality (UoF) should there be more than one UoF defined in the AM. Each row is divided into five columns that provide the details of the mapping. The headings of the five columns are "Application element", "MIM element", "Source", "Rules", and "Reference path". The following subclauses provide guidance on the development of each column of the mapping table.

5.6.1 Application element column documentation

The Application element column lists the application objects (i.e. entities and attributes) and relationship attributes from clause 4 of the AM in accordance with the guidance provided in the *Supplementary directives for the drafting and presentation of ISO 10303*. Each application element from the application module appears at least once in the table. An application object may appear in more than one UoF within an AM. When this occurs, the sub-table for each UoF documents the mapping of that element within the context of that UoF. If the content of the application element column is longer than the column is wide, the line shall be broken between words or following an underscore. Hyphens shall be used where appropriate when a line is broken to provide consistency with the rest of the AM document.

The requirements of the application module may define more than one relationship attribute between two application objects. When this occurs, each relationship attribute is entered on a separate row of the mapping table. The heading of the relationship attribute from the AM is used for each entry along with an identifying phrase. The phrase is chosen from the normative text of the relationship attribute and is placed in parentheses following the relationship attribute heading. The relationship attributes appear in the mapping table in the order that they are defined in the AM. Figure 2 illustrates how three relationship attributes between a pair of application objects are documented in the first column of a mapping table.

4.3.67 Structured_dimension_callout to Text_string

Each Structured_dimension_callout has as a dimension value one or more Text_string objects. Each Text_-string may be the dimension value for exactly one Structured_dimension_callout.

Each Structured_dimension_callout has as a tolerance value zero, one, or many Text_string objects. Each Text_string may be the tolerance value for exactly one Structured_dimension_callout.

Each Structured_dimension_callout has as unit text zero, one, or many Text_string objects. Each Text_string may be the unit text for exactly one Structured_dimension_callout.

Application element	
STRUCTURED_DIMENSION_CALLOUT	
structured_dimension_callout to text_string (as dimension value)	
structured_dimension_callout to text_string (as tolerance value)	
structured_dimension_callout to text_string (as unit text)	

Figure 2 - Example of multiple relationship attributes

Application element	MIM element	Source
ANNOTATION_SUBFIGURE_-DEFINITION_ELEMENT	#1: (draughting_annotation_-occurrence)	201
#1: If the element is a curve, fill area, symbol, subfigure, or text #2: If the element is a dimension or a draughting callout	#2: (draughting_elements)	201

Figure 3 - Example of alternative mappings

When an application object maps to different MIM objects in different contexts, often the mappings can be clearly documented only through the use of numbered alternatives within the row of the mapping table. In these cases, the application element column contains numbered descriptions that indicate when the alternative mappings apply. In the succeeding columns of the table, the MIM elements and reference paths are numbered correspondingly. In practice, when there are multiple mappings for an application object, these mappings are frequently self-explanatory. However, mappings of relationship attributes between objects with multiple mappings generally require the numbered descriptions. Figure 3 shows alternative mappings for an application object that requires clarification. In the example in annex B, the entity Organization has multiple mappings, though numbered alternatives are only provided for the mappings of the Organization_name attribute and the relationship attribute from Approval to Organization.

NOTE - There is a recognized deficiency in the mapping table syntax: the cardinalities and inheritance documented in the information requirements of clause 4 of the AM are not visible in the mapping table. The intent is that the inheritance in the ARM is preserved in the mapping, though the current mapping table syntax provides no means to explicitly show this. For example, the mapping of an relationship attribute from a supertype to another application element would apply also to subtypes of that supertype. Readers of the mapping

table must look to clause 4 of the AM for the cardinalities and subtype/supertype relationships among the application elements.

5.6.2 MIM element column documentation

The MIM element column contains the description of that to which the application element from the preceding column maps. Application objects map to either a single MIM element, a combination of multiple MIM elements, or a choice among multiple MIM elements. MIM elements for application objects are entities or attributes from resource models (i.e., integrated resources or application interpreted constructs) or entities that are defined in the MIM of the AM being mapped. The MIM element column for an relationship attribute is populated with the word "PATH" or the words "IDENTICAL MAPPING". These mappings are described in the following sub-clauses.

The content of the MIM element column is identified during the application interpretation process. This process is based on human understanding of the information requirements presented in the application module and the ISO 10303 integrated resources. This process is described in 5.3.1 of this document.

The MIM element chosen for the mapping shall be as specific as possible. Mappings shall be shown to the attribute that will actually be populated with the data rather than to the entity itself. When the mapping is to an entity, the entity name appears in the MIM element column. When the mapping is to an attribute, the MIM element column contains the entity name where the attribute is defined, followed by a period, followed by the attribute name. If the MIM element name is longer than the column is wide, the MIM element name shall be separated at the period (if there is one) or following an underscore. Hyphens shall be used to provide consistency with the rest of the AM document.

When a supertype is provided as the MIM element for a mapping, the intent is that the application object may map to the supertype itself or any of the subtypes that are within the scope of the MIM of the application module. If the intent is to map to a subset of the subtypes, the MIM element must be specified as a complex mapping (as described in 5.6.2.2) that indicates which subtypes are allowable. If the intent is to map the application object to the supertype only, the name of the supertype entity is surrounded by vertical bars "| |".

5.6.2.1 Single MIM element

A single MIM element is provided in the MIM element column when only one entity or attribute from the integrated resources, application interpreted constructs, or MIM short form maps to the application object.

5.6.2.2 Multiple MIM elements

Sometimes a single application object maps to more than one MIM element. When this occurs, each mapping is documented in the mapping table. This may occur as an "and" situation (both MIM elements must be present), as an "or" situation (either MIM element may be present), or as an "and/or" situation (one or more of the MIM elements must be present). The MIM elements provided in the mapping shall be as specific as possible. For instance, if the data that is to be populated is a combination of a unit and a

value, it is more clear if the mapping is to both measure_with_unit.unit_component and measure_with_unit.value_component than simply to measure_with_unit.

In an "and" mapping, multiple MIM elements are required to satisfy the information requirement. Square brackets "[ ]" are placed around each required MIM element to denote the "and" situation. In an "or" situation, multiple MIM elements are alternatives that satisfy the information requirements. Parentheses "( )" are placed around each alternative MIM element to denote the "or" situation. In an "and/or" situation, at least one of the listed MIM elements is required to satisfy the information requirement. Angle brackets "< >" are placed around each alternative in the "and/or" situation. Parentheses, square and angle brackets may be used in combination in the specification of an MIM element to indicate the mapping of complex logical situations.

The alternatives in complex mappings (typically "or" situations) may be numbered to reflect descriptions provided in the application element column (see clause 5.6.1). These descriptions are provided to clarify the situations under which the mappings apply. The use of numbered alternatives is not mandated, but is left to the discretion of the AM team. If the application of the different mappings is not clear, descriptions shall be provided. For an example of an attribute with multiple mappings, see the Organization_name attribute of the Organization application object in annex B. In this mapping, how the requirement for a name is satisfied depends on whether the organization is satisfied as a person, an organization, or a person within an organization and is thus indicated by parentheses. When a person within an organization is required, the name is satisfied by both the identification of the person and the name of the organization as indicated by the square brackets.

When an application element maps to the same MIM element but from different sources, the name of the MIM element appears more than one time in the MIM element column as described above. In figure 4 the application object Point maps to the MIM entity **point** in the generic context defined by the integrated resource part and also in the context of an application interpreted construct. The entity name appears twice in the MIM element column and the appropriate part number is referenced in the Source column.

Application element	MIM element	Source
POINT #1: If the point is not part of a shape_representation #2: If the point is part of an elementary_brep_shape_-representation	#1: (point) #2: (point)	42 513

Figure 4 - Example of mapping to the same entity from different sources

5.6.2.3 Path

The MIM element column of a mapping table contains the word "PATH" when the application element is an relationship attribute and the application objects in the relationship attribute map to different MIM elements. The reference path for an relationship attribute is designed to show how the relationship between the application objects is satisfied in the MIM.

There are cases where one of the application objects participating in an relationship attribute maps to a complex logical relationship of MIM elements and the other maps to a subset of those MIM elements. Because the reader of the mapping table cannot tell whether this was done on purpose or is an omission, when this case occurs, the MIM element for the relationship attribute is still "PATH", but the mapping shall be clarified with a note so it is clear that the reference path is not incomplete. The note clarifying the mapping shall be referenced in the MIM element column where it applies and the text of the note provided at the end of the mapping table in a notes list. The notes list shall precede the list of rules. The note shall be of the format:

NOTE 1 - For the purpose of this mapping, only the subset of the mapping of the <"to"_application_object> specified in the reference path is applicable.

Drawing_sheet to Sheet_placed_annotation relationship attribute is an example of such an relationship attribute (see figure 5). The relationship between Drawing_sheet and Sheet_placed_annotation does not apply to case two where the annotation is a dimension or a draughting callout. In the case of this example, the note would read:

NOTE 2 - For the purpose of this mapping, only the subset of the mapping of the Sheet_placed_annotation specified in the reference path is applicable.

Application element	MIM element	...	Reference Path
DRAWING_SHEET	dawing_sheet_revision		
drawing_sheet to sheet_placed_annotation	PATH (see NOTE 2)		drawing_sheet_revision <= presentation_area <= presentation_representation <= representation representation.items[i] -> representation_item => styled_item => annotation_occurrence => draughting_annotation_occurrence
SHEET_PLACED_ANNOTATION #1: If the annotation is a curve, fill area, symbol, subfigure, or text #2: If the annotation is a dimension or a draughting callout	#1: (draughting_annotation_-occurrence) #2: (draughting_elements)		#1: (draughting_annotation_occurrence <= annotation_occurrence) #2: (draughting_elements <= draughting_callout)

Figure 5 - Example of PATH MIM element requiring clarification

5.6.2.4 Identical mapping

The MIM element column of a mapping contains the words "IDENTICAL MAPPING" when the application element is an relationship attribute and both application objects in the relationship attribute map to the same MIM element instance. In the example in figure 6, 7 Annotation_subfigure_definition_element maps to **draughting_annotation_occurrence** or to **draughting_elements**. Draughting_annotation maps to the same two choices. The relationship attribute between these two application objects is an "IDENTICAL MAPPING" because these two application objects map to the same instance. Such cases may arise when

there is a high level of detail in the application reference model (ARM) or the constructs in the integrated resources are more semantically rich than the constructs in the ARM.

Application element	MIM element	Source
ANNOTATION_SUBFIGURE_-DEFINITION_ELEMENT #1: If the element is a curve, fill area, symbol, subfigure, or text #2: If the element is a dimension or a draughting callout	#1: (draughting_annotation_-occurrence) #2: (draughting_elements)	201 201
annotation_subfigure_-definition_element to draughting_annotation	IDENTICAL MAPPING	
DRAUGHTING_ANNOTATION #1: If the annotation is a curve, fill area, symbol, subfigure, or text #2: If the annotation is a dimension or a draughting callout	#1: (draughting_annotation_-occurrence) #2: (draughting_elements)	201 201

Figure 6 - Example of an identical mapping

When one of the application objects participating in an relationship attribute maps to a complex logical relationship of MIM elements and the other maps to a subset of those MIM elements, the MIM element for the relationship attribute is still IDENTICAL MAPPING, and the mapping may be clarified with a general note stating that only the intersection of the MIM elements to which the two objects map are the same instance. The note clarifying the mapping, if it is deemed necessary, shall be referenced in the MIM element column in as many places as it applies and the text of the note shall be provided at the end of the mapping table in a notes list. The notes list shall precede the list of rules. The note shall be of the format:

> NOTE - For the purpose of this mapping, only the intersection of the mappings for each of the application objects in the relationship attribute is applicable.

5.6.3 Source column documentation

The Source column contains an ISO 10303 part number for each MIM element provided in the preceding column. In general, the part number is the part of ISO 10303 in which the MIM element is defined. The part numbers referenced in a mapping table may correspond to an integrated resource; an application interpreted construct; or the application module itself, in the case where the MIM element is an AM created specialization of an integrated resource entity. When an application object or relationship attribute is mapped to an entity or type that is defined in the integrated resources, implicitly or explicitly brought into the scope of an AIC according to the interfacing rules of EXPRESS as documented in ISO 10303-11, and the mapping is within the context of the AIC, the AIC part number is referenced in the source column. If the mapping is not within the context of the AIC, the part number of the integrated resource construct is referenced in the source column.

If the MIM element column contains either the word “PATH” or the words “IDENTICAL MAPPING”, no source document is listed in the source column.

5.6.4 Rules column documentation

The Rules column of the mapping table contains numerical references to the global rules in the MIM short form that constrain the mappings. The numerical references correspond to a numbered list of all global rules in the AP. That list follows the mapping table. In this list, the rules shall appear in the same order as in the AM. There may be more than one rule constraining a given mapping. Some mappings may be unconstrained. Rules restricting instantiation of entities within the MIM are to be included in the mapping table only when the mapping is to an MIM element that shall not be independently instantiable. When an entity constrained by an instantiability rule appears in a reference path, that mapping assumes that the entity will be instantiated in the context of other entities; therefore, the reference to the rule is not needed for that mapping. All rules that are created in the AM short form, including entity instantiability rules, shall be referenced in the mapping table at least once.

5.6.5 Reference path column documentation

The reference path illustrates how the requirements and relationships stated in clause 4 of the AM are maintained as a result of the application interpretation process. It specifies the complete path of entity references in the MIM that is needed to represent the information requirements of the ARM. A set of symbols and formats were developed to construct a consistent syntax for documenting reference paths. Reference path syntax is consistent for each type of application element. The intent of the current syntax is to facilitate human readability of the mapping table. In the future, this syntax may be extended to improve computer readability of the mapping table. This clause discusses the symbology used in documenting reference paths as well as reference path requirements for each type of application element.

5.6.5.1 Symbology

A reference path specification is contained within a single cell in the reference path column of the mapping table and is read from top to bottom. Each line of the reference path specification may contain symbols to illustrate the EXPRESS structure of the MIM objects. Understanding the symbology used in the reference path specification is the key to reading and writing mapping tables.

5.6.5.1.1 Delimiter symbols

The delimiter symbols are used to indicate the relationship of the specified entity or attribute preceding the symbol to the specified entity or attribute following the symbol. The symbol should be placed at the end of the line, so that the name of the following entity or attribute is at the beginning of the next line. The delimiters should be separated from the entity or attribute text by a single space, for readability. The meaning of these symbols can be paraphrased as:

=> :	"is a supertype of"
<= :	"is a subtype of"
-> :	"references"
<- :	"is referenced by"

The "=>" and "<=" symbols indicate a supertype or subtype structure. The "=>" symbol is used to indicate that the specified entity preceding the symbol is the supertype of the entity specified on the next

line. The "<=" symbol is used to indicate that the specified entity preceding the symbol is a subtype of the entity specified on the next line.

The "->" and "<-" symbols indicate the reference to an entity or type by an attribute. The "->" symbol is used to indicate that the specified attribute preceding the symbol references the entity or type specified on the next line. The "<-" symbol is used to indicate that the specified entity or type preceding the symbol is referenced by the attribute specified on the next line.

When an entity name appears on a line that is terminated without the use of one of the above delimiter symbols, this may indicate that the specified entity has the attribute shown on the next line. When an attribute name appears on a line that is terminated without the use of one of the above symbols, this may indicate that the specified attribute is an attribute of the entity shown on the next line. A new line is used without conveying additional semantics before and after lines where an EXPRESS select type value is provided (see 5.6.5.1.3). New lines precede and follow mapping rules (braces), and the individual options for mappings with multiple MIM elements (parentheses, square and angle brackets). A new line also terminates the reference path.

Reference path statements that are too long to fit within the table cell can be split using the forward slash symbol "\". The symbol conveys no additional meaning within the reference path. The "\" is positioned between elements of the statement, at the end of the line, preferably in white space. The "\" may appear between an entity and an attribute, following the period, but this case should be avoided where possible. The "\" should not be placed inside of a text string; the entire text string should follow the forward slash on the next line of that cell.

5.6.5.1.2 Aggregation symbols

If an attribute references an aggregate cardinality, and any single instance of the aggregate is of interest, brackets and the letter i "[i]" are used to indicate this. This reflects the usual requirement that the path can go through any member of the aggregate. The use of "[n]" (where n is an integer) indicates that member n of the aggregate is of interest in the mapping. A reference to an_entity.aggregate[1] reflects the requirement that the path must go through the first element of the aggregate. In order to limit the number of elements in an instantiation of the aggregation, EXPRESS rules shall be written in the short form of the MIM. In the mapping of the Drawing to Approval relationship attribute in annex B, the attribute **approved_items** is a set that references the select type **approved_item**.

5.6.5.1.3 Equal sign

An equal sign "=" is used in the reference path specification to indicate a member of the select list of an EXPRESS select type, an item from the enumerated list of an EXPRESS enumeration type, or a specific value for an attribute. In the case of a select list, the name of the select type appears first followed on the same line by the equal sign and the member that is being selected. In the case of an enumerated list, the name of the enumeration type appears first followed on the same line by the equal sign and the enumerated item. In the case of a specific value, the attribute name appears first followed on the same line by the equal sign and the value assigned to the attribute.

See the mapping of the Date attribute of the Approval object in annex B. In this reference path specification, the attribute **date_time** references the select type **date_time_select**. For this mapping, the selection is a **date**. As seen in this example, the name of the select type appears on the line before the line containing the equal sign, and the selected member appears on the line following the line containing the equal sign. The order of these lines is reversed for a reference path in which the selected member is referenced first in the path. See the mapping of the Drawing to Approval relationship attribute in the example in annex B. In this mapping, the reference path encounters **drawing_revision** first, which is the selected member of the **approved_item** select type.

5.6.5.1.4 Parentheses

Parentheses "( )" are used to indicate the existence of options in the reference path. Each option is enclosed by a set of parentheses. The parentheses are used to indicate that a mapping has multiple reference paths or sections of the reference path. There are two reasons that the reference path may diverge: an object is mapped to multiple MIM entities or the reference path depends on the instantiation of the MIM. To aid understanding, the optional sections of the path may be numbered and a description provided, with the application object, that gives the reason for the divergence.

See the mapping of the Approval to Organization relationship attribute in annex B. In this example, it is necessary to show the reference paths for each of the MIM elements to which the Organization maps.

5.6.5.1.5 Square brackets

Square brackets "[ ]" are used to indicate two or more required sections of the reference path. The square brackets indicate that there are either multiple mappings or multiple paths required to satisfy the mapping. Each mapping or path is enclosed by a set of square brackets. To fully document how the requirements are satisfied by the mapping, sections of the path may be numbered and a description giving the reason for the divergence may be provided in the Application element column.

See the mapping of the Annotation_subfigure_definition to 2D_cartesian_coordinate_space relationship attribute in the example in figure 7. This example shows that every **representation_context** that satisfies the requirements of the application object 2D_cartesian_coordinate_space must be both a **geometric_representation_context** and a **global_unit_assigned_context**.

Application element	MIM element	Source		Reference path
ANNOTATION_SUBFIGURE_- DEFINITION	symbol_representation_- map	46		{symbol_representation_map <= representation_map representation_map.mapped_representation ->> representation => symbol_representation => draughting_subfigure_representation}
annotation_subfigure_definition to 2d_cartesian_coordinate_space	PATH			symbol_representation_map <= representation_map representation_map.mapped_representation ->> representation {representation => symbol_representation => draughting_subfigure_representation} representation.context_of_items ->> representation_context => [geometric_representation_context] [global_unit_assigned_context]
2D_CARTESIAN_- COORDINATE_SPACE	[geometric_representation_- context] [global_unit_assigned_- context]	42 41		

Figure 7 - Example of multiple mappings

NOTE - 9.2 of ISO 10303-11 enumerates the type of constraints that can be written for EXPRESS entities.

Mapping rules are used when the reference path specification would not normally contain MIM entities which are crucial to the mapping. Mapping rules may be used to include into the reference path specification required supertypes or subtypes, required values assigned to attributes, or AIC entities containing rules that constrain the mapping.

Mapping rules shall start on a new line of a reference path and shall end with a new line. The reference path shall be completely specified without the mapping rule; if the mapping rule were removed, the reference path would still be correct. If a reference path contains multiple mapping rules, all the rules apply to the mapping. If the granularity of the ARM and MIM differ such that the resulting mapping rules apply to different cases, use numbered alternatives in the ARM element column and number the mapping rules correspondingly in the reference path column.

Mapping rules are also used when an assertion is mapped to a specialization of a resource entity and the reference path specification would not otherwise show the resource entity. The inclusion of these MIM elements is intended to satisfy the requirement that all mappings to a specialized entity shall have a reference path to a resource entity.

The mapping of the Structured_dimension_callout to Draughting_callout relationship attribute in the example in figure 8 contains two mapping rules. In this example, the reference path specification would not

Application element	MIM element	Source		Reference path
STRUCTURED_DIMENSION_-CALLOUT	structured_dimension_-callout	201		structured_dimension_callout <= draughting_callout
structured_dimension_callout to draughting_callout (as prefix)	PATH			structured_dimension_callout <- draughting_callout_relationship.relating_dimension_callout draughting_callout_relationship {draughting_callout_relationship <= dimension_callout_component_relationship} {draughting_callout_relationship.name = 'prefix'} draughting_callout_relationship.related_draughting_callout -> draughting_callout
DRAUGHTING_CALLOUT	draughting_callout	101		

Figure 8 - Example of mapping rules

normally show the ISO 10303-201 created subtype. However, for this mapping, the only **draughting_callout_relationship** that satisfies the information requirement is the **dimension_callout_component_relationship** subtype. The second mapping rule indicates that the name of the relationship is restricted for this mapping to have the value "prefix".

Another type of mapping rule is used to signify that any number of relationship entities may be assembled in a relationship tree structure. This is shown by the use of the relationship entity name followed by an asterisk "*". There should be white space between the entity name and the asterisk. An asterisk may only be used in a mapping rule, and the only currently known application of this syntax is in extending relationship trees to any number of levels. In the absence of a mapping rule including an asterisk, the path contains only the number of relationship entities shown. In the following example reference path there may be any number of **draughting_callout_relationship** entities in the relationship tree.

```
draughting_callout <-
draughting_callout_relationship.related_draughting_callout
draughting_callout_relationship
{draughting_callout_relationship *}
draughting_callout_relationship.relating_draughting_callout ->
draughting_callout
```

5.6.5.1.6 Vertical bars

Vertical bars "| |" are used to indicate that the mapping is only to the supertype entity found between the vertical bars. If the vertical bars are not present, and the entity being referenced is a supertype, it is assumed that any of the subtypes of that supertype are valid in the context of that reference path. The vertical bars limit the reference path to the supertype entity itself. Vertical bars may appear in both the MIM element column and the reference path.

```
A.b ->
| only_supertype_allowed |
```

5.6.5.1.7 Single quotes

Single quotes " ' ' " are used to indicate that the enclosed text is intended to be the value of an attribute whose underlying type is a string. Single quotes are used in this context to provide consistency with ISO 10303-11. They may appear in the reference path only.

5.6.5.2 Application objects

A reference path specification is necessary for each application object that is mapped to a specialization of an integrated resource entity. The reference path starts with the MIM element to which the application object is mapped. It concludes at the integrated resource entity of which the specialization is a subtype. See the mapping of Drawing in annex B. In this example, Drawing is mapped to **draughting_drawing_revision**, which is a subtype of **drawing_revision** in ISO 10303-101. The reference path contains these two entities.

A reference path can also be shown to clarify a restriction on the mapping. See the mapping of Annotation_subfigure_definition in the example in figure 6. This object maps to an entity in the integrated resources; therefore, no reference path specification is required. However, to satisfy the requirements for this mapping, the inherited attribute **mapped_representation** must be of type **draughting_subfigure_-representation**. This restriction is shown by including this portion of the reference path specification within braces (see 5.6.5.1.7).

5.6.5.3 Application attributes

There are different mappings of an attribute that must be considered for the documentation of the reference path. An attribute may be mapped to an attribute of the same MIM entity to which the application object is mapped. In this case, no reference path is necessary for the attribute. See the mapping of the Description attribute of Approval in the example in annex B.

An attribute may be mapped to an attribute of an entity different from the one to which the application object is mapped. The reference path for the attribute starts with the MIM element to which the application object is mapped. The path follows the entities and attributes of the MIM to the MIM attribute to which the application attribute is mapped.

See the mapping of the Drawing_number attribute of Drawing in the example in annex B. In this example, Drawing is mapped to **draughting_drawing_revision** so the reference path of Drawing_number begins with this entity. **Drawing_revision** has an attribute **drawing_identifier** that references the entity **drawing_definition**. The **drawing_definition** entity has an attribute **drawing_number** to which the Drawing_number attribute is being mapped.

5.6.5.4 Relationship attributes

Relationship attributes specify the relationships between pairs of application objects, the cardinality of the relationships, and the rules required for the integrity and validity of the application objects.

The reference path of an relationship attribute starts with the MIM element to which the first application object is mapped. The path concludes at the MIM element to which the second application object in the relationship attribute is mapped. See the mapping of the Drawing to Approval relationship attribute in annex B. In this example, Drawing is mapped to **draughting_drawing_revision** so the reference path of the relationship attribute begins with this entity. **Drawing_revision** is one of the **approved_items** to which **approval** is assigned.

If the MIM element column contains the words “IDENTICAL MAPPING”, no reference path specification is required.

In rare cases, an relationship attribute may map to an MIM entity or attribute. In these cases, the mapping may be to an attribute in the reference path that connects the two identified application objects, or to an MIM entity that acts as the intersection entity connecting the two identified application objects. This attribute or entity is selected for the MIM element column with agreement by the MIM interpretation committee. The source column contains the number of the part containing this entity or attribute.

5.6.5.5 AIC considerations

AICs define entities where the global rules pertaining to that AIC have been localized. These entities are called "root node" entities. If the root node of the AIC includes restrictions that apply to the mapping, the root node shall be included in the reference path specification to indicate this, even if the inclusion is only within a mapping rule.

Annex A

Example information requirements

This annex contains example descriptions of a UoF, application objects, and relationship attributes that appear in the example mapping table in annex B. These descriptions are extracted from clause 4 of ISO 10303-201. Some objects, attributes, and relationship attributes from the UoF have been excluded from the example.

NOTE - The numbering in this annex reflects the clause numbering as published in ISO 10303-201.

4.1 Units of functionality

4.1.4 drawing_structure_and_administration

The drawing_structure_and_administration UoF contains information about the hierarchical organization of drawings, drawing sheets, and drawing views, together with the administrative information necessary to manage drawings and drawing sheets. Drawing sheets and drawing views are defined in their specific coordinate space. Annotation may be assigned to each drawing sheet and drawing view. The administrative information supports the exchange of drawings between environments in which configuration management of drawings is used. The following application objects are used by drawing_structure_and_administration UoF:

— Approval;

— Drawing;

— Drawing_sheet;

— Organization.

4.2 Application objects

4.2.13 Approval

An Approval is information that indicates a drawing, drawing sheet, or both have been reviewed for data content and for correctness of the presentation of that data and has been found to be acceptable. The data associated with an Approval are the following:

— Date;

— Description.

4.2.19.1 Date

The Date specifies the date on which the approval was assigned.

4.2.19.2 Description

The Description specifies the organization-specific release status or the authorized modifications for the revision of the drawing, drawing sheet, or both.

4.2.30 Drawing

A Drawing is the presentation of product data in a human-interpretable form wherein the physical and functional requirements for that product are presented pictorially and textually. The data associated with a Drawing are the following:

— Drawing_number;

— Drawing_revision_id.

4.2.30.2 Drawing_number

The Drawing_number specifies the identification of a particular drawing by an organization.

4.2.30.3 Drawing_revision_id

The Drawing_revision_id specifies the identification of a particular version of the drawing.

4.2.31 Drawing_sheet

A Drawing_sheet is a logical division of a drawing into a two-dimensional area for the presentation of product data. These divisions correspond to sheet paper sizes for plotting. A Drawing_sheet contains at least one Drawing_view or one Draughting_annotation. The data associated with a Drawing_sheet are the following:

— Sheet_number;

— Sheet_revision_id.

4.2.31.2 Sheet_number

The Sheet_number specifies the page number for a particular drawing sheet and its location in relation to other sheets of the drawing.

4.2.31.3 Sheet_revision_id

The Sheet_revision_id specifies the identification of a particular version of the drawing sheet.

4.2.57 Organization

An Organization is a number of persons or groups that designs, produces and supplies products and services. The data associated with an Organization are the following:

— Organization_name.

4.2.57.2 Organization_name

The Organization_name specifies the identification of a particular organization.

4.3 Relationship attributes

4.3.18 Approval to Organization

Each Approval is provided by one or more Organization objects. Each Organization provides zero, one, or many Approval objects.

4.3.32 Drawing to Approval

Each Drawing is governed by zero, one, or many Approval objects. Each Approval governs zero or one Drawing.

4.3.33 Drawing to Drawing_sheet

Each Drawing consists of one or more Drawing_sheet objects. Each Drawing_sheet belongs to exactly one Drawing.

4.3.37 Drawing_sheet to Approval

Each Drawing_sheet is governed by zero, one, or many Approval objects. Each Approval governs zero, one, or many Drawing_sheet objects.

Annex B

Example mapping table

This annex contains the mapping table that corresponds to the example information requirements in annex A. See annex A for the textual descriptions of the application objects.

The MIM entities found in this mapping table are defined in ISO 10303-41 [4], ISO 10303-101 and ISO 10303-201.

Table B.1 - Mapping table for drawing_structure_and_administration UoF

Application element	MIM element	Source	Rules	Reference path
APPROVAL	approval	41		
date	calendar_date	41		approval <- approval_date_time.dated_approval approval_date_time approval_date_time.date_time ->>> date_time_select date_time_select = date date => calendar_date
description	approval.level	41		
approval to organization #1: If the approval is given by only a person #2: If the approval is given by only an organization #3: If the approval is given by a person within an organization	PATH			approval <- approval_person_organization.authorized_approval approval_person_organization approval_person_organization.person_organization ->>> person_organization_select #1: (person_organization_select = person person) #2: (person_organization_select = organization organization) #3: (person_organization_select = person_and_organization person_and_organization)

Table B.1 - Mapping table for drawing_structure_and_administration UoF (continued)

Application element	MIM element	Source	Rules	Reference path
DRAWING	draughting_drawing_-revision	201		draughting_drawing_revision <= drawing_revision
drawing_number	drawing_definition. drawing_number	101		draughting_drawing_revision <= drawing_revision drawing_revision.drawing_identifier -> drawing_definition drawing_definition.drawing_number
drawing_revision_id	drawing_revision. revision_identifier	101		draughting_drawing_revision <= drawing_revision drawing_revision.revision_identifier
drawing to approval	PATH			draughting_drawing_revision <= drawing_revision approved_item = drawing_revision approved_item <- draughting_approval_assignment.approved_items[i] draughting_approval_assignment <= approval_assignment approval_assignment.assigned_approval -> approval
drawing to drawing_sheet	PATH			draughting_drawing_revision <= drawing_revision <= presentation_set <- area_in_set.in_set area_in_set {area_in_set => drawing_sheet_revision_usage} area_in_set.area -> presentation_area => drawing_sheet_revision

Table B.1 - Mapping table for drawing_structure_and_administration UoF (concluded)

Application element	MIM element	Source	Rules	Reference path
DRAWING_SHEET	drawing_sheet_revision	101		
sheet_number	drawing_sheet_- revision_usage. sheet_number	101		drawing_sheet_revision <= presentation_area <- area_in_set.area area_in_set => drawing_sheet_revision_usage drawing_sheet_revision_usage.sheet_number
sheet_revision_id	drawing_sheet_revision. revision_identifier	101		
drawing_sheet to approval	PATH			drawing_sheet_revision approved_item = drawing_sheet_revision approved_item <- draughting_approval_assignment.approved_items[i] draughting_approval_assignment <= approval_assignment approval_assignment.assigned_approval ->approval
ORGANIZATION	(person) (organization) (person_and_- organization)	41 41 41		
organization_name #1: If the organization is only a person #2: If the organization is only an organization #3: If the organization is a person within an organization	#1: (person.id) #2: (organization.name) #3: ([person.id] [organization.name])	41 41 41 41		#3: (person_and_organization [person_and_organization.the_person -> person person.id] [person_and_organization.the_organization -> organization organization.name])

Annex C

Constraint templates

C.1 Dependent instantiation

This template may be used when `entity_a` may only be instantiated as referenced by another entity.

```
*)
RULE dependent_instantiation_entity_a FOR (entity_a);
WHERE
  WR1:  SIZEOF (QUERY (each <* entity_a |
        NOT (SIZEOF (USEDIN (each, ‘’)) >=1
        ))) =0;
END_RULE;
(*
```

C.2 Mandatory subtype

This template may be used when a supertype (`entity_a`) is constrained to be one of more than one valid subtypes.

```
*)
RULE subtype_mandatory_entity_a FOR (entity_a);
WHERE
  WR1: SIZEOF (QUERY (temp_a <* entity_a |
       NOT (SIZEOF (TYPEOF (temp_a)*
         [’SCHEMA_NAME.VALID_SUBTYPE_1',
         ‘SCHEMA_NAME.VALID_SUBTYPE_2']
           )=1))) =0;
END_RULE;
(*
```

This template may be used when the supertype entity is constrained to be one specific subtype.

```
*)
RULE subtype_mandatory_entity_a FOR (entity_a);
WHERE
   WR1: SIZEOF (QUERY (temp_a <* entity_a |
         NOT (’SCHEMA_NAME.VALID_SUBTYPE' IN TYPEOF (temp_a)))) =0;
END_RULE;
(*
```

C.3 Cardinality

This template may be used when `entity_b` has an attribute `x` that references `entity_a`. This rule constrains the relationship to a cardinality other than the default, zero, one or many.

```
*)
RULE entity_a_to_entity_b FOR (entity_a, entity_b);
WHERE
   WR1: SIZEOF (QUERY (temp_a <* entity_a |
         NOT (SIZEOF (QUERY (temp_b <* entity_b |
         temp_a :=: temp_b.x)) =1))) =0;
END_RULE;
(*
```

The desired cardinality is reflected by the cardinality to which the interior SIZEOF statement is compared. In the template, the cardinality is exactly one. To constrain the cardinality to be greater than or equal to one, " >=1" could be substituted for the "=1", for example.

If attribute `x` of `entity_b` is a SET of entity_a, use the IN operator instead of the entity instance equality operator. If attribute `x` of `entity_b` is a BAG, LIST or ARRAY of non-unique `entity_a`, the cardinality of a single instance of `entity_b` to a single instance of `entity_a` needs to be considered and isn't covered by this template.

C.4 Value restriction

This template may be used when `entity_a` has an `attribute_b` that is of type STRING. The values in single quotes are the allowable values of `attribute_b`. If there is only one allowable value for `attribute_b`, use the equal sign in place of the IN operator.

```
*)
RULE restrict_value_of_atribute_b FOR (entity_a);
WHERE
   WR1: SIZEOF (QUERY (temp_a <* entity_a |
         NOT (attribute_b IN ['string 1', 'string 2', 'string 3']))) = 0;
END_RULE;
(*
```

Annex D

Steps for writing MIM short form EXPRESS

1. Develop the EXPRESS definitions for all select types, subtype entities (including local rules), global rules and functions identified during the interpretation workshops. Develop EXPRESS global rules for constraints documented as mapping rules within the mapping table where possible.

2. Write USE FROM statements for all foreign-defined named types (entities and types) that are needed to support the creation of MIM unique constructs from step 1. See the discussions on visibility in ISO 10303-11:1994 to understand EXPRESS visibility requirements. Any foreign construct that is named in the definition of a local construct, with the exception of those named only in strings, shall be made visible through inclusion in a USE statement.

3. Write USE FROM statements for each AM (and AIC). In some cases, an AM (or AIC) may contain USE FROM statements for a construct identified in step 2. Because an AM is a separately balloted document, do not depend on an or AIC to interface constructs determined in step 2 to support MIM short form locally defined constructs. In this case, redundant USE FROM statements are allowed.

   NOTE TO STEP A/F TEAM - We need to capture the idea of referencing constructs from modules, AICs, IRs and what the differences are between the different references.

4. Examine the integrated resource EXPRESS-G to determine which constructs will be implicitly interfaced based on the explicit interfaces written in steps 2 and 3. See the discussions in ISO 10303-11:1994 to understand what is implicitly interfaced for each explicitly interfaced construct. This includes all the entities supertypes, the named types referenced by the entity's attributes, and SELECT types that contain explicitly or implicitly interfaced constructs in their select lists. Items in select lists are NOT interfaced when the SELECT type is interfaced.

5. Look at the mapping table for additional entities that must be brought in to satisfy reference paths. Each entity that appears in a reference path must appear in the MIM long form (i.e., must be locally defined, explicitly or implicitly interfaced.) Add USE FROM statements to bring in required constructs. When writing the additional USE FROM statements, try to minimize the number of entities named, avoiding, where possible, explicitly interfacing entities that are not intended to be independently instantiated.

6. Examine USE FROM statements for entities that are not allowed to be independently instantiated. Write rules constraining the instantiation of these entities. See constraint templates in annex B of this document for assistance. Entities that are subtypes of representation_item or its subtypes do not require dependent instantiation rules as such a constraint is locally defined in representation_item and inherited by all subtypes. As they cannot be independently instantiated anyway, provide only the minimal set of use statements, i.e., explicitly interface only the bottom leaf subtypes.

   NOTE - Some entities are independent by definition. These must be explicitly interfaced to bring them into the MIM. They are not referenced by other entities and it is, therefore, incorrect to write a dependent instantiation rule for such entities.

7. Add USE FROM statements for any entities that are intended to be independent but that were not already explicitly interfaced in the execution of a previous step. Some constructs may be implicitly interfaced that are required to be independently instantiable. These entities must be explicitly interfaced. Independent entities are those entities that may be instantiated without serving in the role of another entity's attribute.

Annex E

Bibliography

[1] ISO 10303-41:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 41: Integrated generic resources: Fundamentals of product description and support.*

[2] ISO 10303-42:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 42: Integrated generic resources: Geometric and topological representation.*

[3] ISO 10303-43:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 43: Integrated generic resources: Representation structures.*

[4] ISO 10303-202:1996, *Industrial automation systems and integration — Product data representation and exchange — Part 202: Application protocol: Associative draughting*.

[5] ISO 10303-203:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 203: Application protocol: Configuration controlled 3D designs of mechanical parts and assemblies.*

[6] *ISO TC 184/SC4 organization handbook*, ISO TC 184/SC4 N 492, October 1996.